



CAMPBELL
STEPHENSON
ASCOLESE LLP

4807 Spicewood Springs Road
Building 4, Suite 201
Austin, Texas 78759
T: 512-439-5080
F: 512-439-5099

Atty. Docket No.: CIS0046US

March 9, 2006

Mail Stop Appeal Brief - Patents
COMMISSIONER FOR PATENTS
P. O. Box 1450
ALEXANDRIA, VA 22313-1450

Re: Applicants: Faisal Haq; Hari K. Lalgudi
Assignee: Cisco Technology, Inc.
Title: IMPLEMENTING ACCESS CONTROL LISTS USING A BALANCED HASH
TABLE OF ACCESS CONTROL LIST BINARY COMPARISON TREES
Serial No.: 09/483,110
Examiner: Frank Doung
Docket No.: CIS0046US

Filed: January 14, 2000
Group Art Unit: 2666

Dear Sir:

Transmitted herewith are the following documents in the above-identified application:

- (1) Return Receipt Postcard;
- (2) This Transmittal Letter (1 page) (*in duplicate*); and
- (3) Appeal Brief (28 pages).

- ☐ No additional fee is required.
☒ The fee has been calculated as shown below:

- | | |
|--|-----------|
| <input checked="" type="checkbox"/> Fee Under 37 CFR § 1.17(f) for Filing an Appeal Brief | \$ 500.00 |
| <input type="checkbox"/> Fee for Petition for Extension of Time (<i>X Month</i>) | \$ 0.00 |
| <input checked="" type="checkbox"/> Conditional Petition for Extension of Time: If an extension of time is required for timely filing of the enclosed document(s) after all papers filed with this transmittal have been considered, an extension of time is hereby requested. | |
| <input checked="" type="checkbox"/> Please charge our Deposit Account No. 502306 in the amount of | \$ 500.00 |
| <input checked="" type="checkbox"/> Also, charge any additional fees required and credit any overpayment to our Deposit Account No. 502306. | |

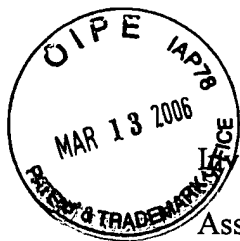
TOTAL \$ 500.00

I hereby certify that this correspondence is being deposited with the United States Postal Service as First Class Mail in an envelope addressed to: Mail Stop Appeal Brief - Patents, Commissioner for Patents, P. O. Box 1450, Alexandria, Virginia, 22313-1450, on March 9, 2006.

Brenna A. Brock 3-9-2006
Attorney for Applicant Date of Signature

Respectfully submitted,

Brenna A. Brock
Brenna A. Brock
Attorney for Applicant
Reg. No. 48,509
Telephone: (512) 439-5087
Facsimile: (512) 439-5099



PATENT

AF
ZFWIN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Inventors: Faisal Haq; Hari K. Lalgudi
Assignee: Cisco Technology, Inc.
Title: Implementing Access Control Lists Using A Balanced Hash Table of
Access Control List Binary Comparison Trees
Serial No.: 09/483,110 Filing Date: January 14, 2000
Examiner: Frank Doung Group Art Unit: 2666
Docket No.: CIS0046US

Austin, Texas
March 9, 2006

MAIL STOP: APPEAL BRIEF - PATENTS
COMMISSIONER FOR PATENTS
P. O. BO 1450
Alexandria, VA 22313-1450

APPEAL BRIEF

Dear Sir:

This brief is submitted in support of the Notice of Appeal filed on December 22, 2005, by Appellant to the Board of Patent Appeals and Interferences from the Examiner's final rejection of Claims 27-30, 32-36, 38-43, 45-49 and 51-59. A Notice of Panel Decision from Pre-Appeal Brief Review was mailed on February 9, 2006. This brief is timely submitted within the one-month period, expiring March 9, 2006, after the mailing of the Notice of Panel Decision from Pre-Appeal Brief Review.

Please charge deposit account No. 502306 for the fee of \$500.00 associated with this appeal brief. Please charge this deposit account for any additional sums which may be required to be paid as part of this appeal.

03/14/2006 TBESHAH1 00000021 502306 09483110
01 FC:1402 500.00 DA

REAL PARTY IN INTEREST

The real party in interest on this appeal is the Assignee, Cisco Technology, Inc.

RELATED APPEALS AND INTERFERENCES

None

STATUS OF CLAIMS

Claims 1-64 are pending in the application.

Claims 1-6, 16-21, 31-43, and 52-57 are rejected.

Claims 7-15, 22-30, 44-51, and 58-64 are objected to as being dependent upon rejected base claims.

Appellant appeals the rejections of claims 1-6, 16-21, 31-43, and 52-57.

STATUS OF AMENDMENTS

No amendments were filed subsequent to the final rejection of August 23, 2005.

SUMMARY OF THE CLAIMED SUBJECT MATTER

Independent claim 1 is directed to a method. The method describes receiving at least one packet. *See, e.g.*, Specification at line 11 and element 112 of FIG. 1A. The method also describes disposing of the received packet in response to a walk of a hash table. *See, e.g.*, Specification at p. 9, lines 15-17 and element 152 of FIG. 1A. The hash table is configured to store Binary Comparison Trees and to encode an Access Control List (ACL). *See, e.g.*, Specification at p. 9, lines 14-17. For example, the hash table can have “a relatively balanced set of entries of ACL binary comparison trees,” “where such constructed hash table will encode the ACL.” *Id.* at p. 8, lines 19-23. The hash table is also balanced. *See, e.g., id.* at pp. 8-9.

Claim 2 depends from claim 1. Claim 2 describes how disposing of the received packet can involve constructing a hash table index value from one or more bit positions

within the received packet. *See, e.g., id.* at p. 9, lines 13-14. Those bit positions within the received packet are pointed at by one or more pointers of a Hash-Table-Balancing Bit Selection Vector. *See, e.g., id.* at p. 9, lines 9-20. Once the hash table index is constructed, a binary comparison tree associated with the constructed hash table index is then walked. *See, e.g., id.*

Independent claim 16 describes a system that includes means for receiving at least one packet. *See, e.g.,* elements 112 and 150 of FIG. 1A as well as elements 150, 128, and 130 of FIG. 1B and Specification at p. 6, lines 18-19 and p. 9, line 9-p. 10, line 2. The system also includes means for disposing of the received at least one packet in response to a walk of a Hash Table. *See, e.g.,* elements 150 and 152 of FIG. 1A as well as element 150 of FIG. 1B and Specification at p. 6, lines 19-22 and p. 9, line 9-p. 10, line 2. As described above, the Hash Table is balanced and is configured to store Binary Comparison Trees as well as to encode an Access Control List. *See, e.g.,* FIGs. 7D1-7D12 and Specification at p. 8, lines 19-23 and p. 9, lines 9-20.

Claim 17 depends from claim 16 and includes additional means for constructing a hash table index value from one or more bit positions, within the received packet, pointed at by one or more pointers of a Hash-Table-Balancing Bit Selection Vector. *See, e.g.,* elements 150 and 152 of FIG. 1A and element 150 of FIG. 1B and Specification at p. 9, lines 9-40. Claim 17 also includes means for walking a binary comparison tree associated with the constructed hash table index value. *See, e.g., id.*

Independent claim 35 describes a program product (*see, e.g.,* Specification at p. 25, lines 12-34) that includes signal bearing media (*see, e.g.,* FIG. 1B and Specification at p. 9, line 31-p. 10, line 2 and at p. 25, lines 29-34) and bearing means for receiving at least one packet. *See, e.g.,* elements 128, 130, and 150 of FIG. 1B. The signal bearing media also bears means for disposing of the received packet in response to a walk of a Hash Table. *See, e.g.,* elements 150 and 152 of FIG. 1A and element 150 of FIG. 1B and Specification at p. 6, lines 19-22 and p. 9, line 9-p. 10, line 2. As described above, the Hash Table is balanced and is configured to store Binary Comparison Trees as well as to encode an Access Control List. *See, e.g.,* Specification at p. 8, lines 19-23 and p. 9, lines 9-20.

Dependent claim 38 depends from claim 35. Claim 38 describes how the means for disposing of the received packet can include means for constructing a hash table index value from one or more bit positions, within the received packet, pointed at by one or more pointers

of a Hash-Table-Balancing Bit Selection Vector. *See, e.g.*, elements 150 and 152 of FIG. 1A and element 150 of FIG. 1B and Specification at p. 9, lines 9-40. The means for disposing of the received packet can also include means for walking a binary comparison tree associated with the constructed hash table index value. *See, e.g., id.*

Independent claim 52 describes a network station that includes a per-packet processing engine (*see, e.g.*, elements 150 and 152 of FIG. 1A and Specification at p. 7, lines 30-35 and p. 9, lines 9-20) and a hash table (*see, e.g.*, element 110 of FIG. 1A and FIGs. 7D1-7D12). The hash table is balanced and is configured to store binary comparison trees and to encode an access control list. *See, e.g.*, Specification at p. 8, lines 19-23 and p. 9, lines 9-20. The per-packet processing engine is configured to walk the hash table, in response to the network station receiving a packet. *See, e.g.*, Specification at p. 9 and elements 112, 150, and 152 of FIG. 1A. The walk of the hash table identifies a disposition of the packet, according to the access control list encoded in the hash table. *See, e.g.*, Specification at p. 9 and element 114 of FIG. 1A.

Dependent claim 53 describes how the per-packet processing engine can be configured to construct a hash table index value from one or more bit positions, within the packet, pointed at by one or more pointers of a hash-table-balancing bit selection vector. *See, e.g.*, Specification at p. 9 and element 152 of FIG. 1A. The per-packet processing engine can also be configured to walk a binary comparison tree, stored in the hash table, associated with the constructed hash table index value. *See, e.g., id.*

GROUND OF REJECTION TO BE REVIEWED ON APPEAL

Claims 1, 3-6, 16, 18-21, 31-37, 39-43, 52, and 54-57 stand rejected under 35 U.S.C. § 102(b), as being anticipated by Dobbins, et al., U.S. Patent No. 5,509,123 (hereinafter referred to as “Dobbins”), as indicated in the Final Office Action mailed August 23, 2005 (hereinafter referred to as “FOA”), the Advisory Action mailed November 14, 2005 (hereinafter referred to as “AA”), and the Notice of Panel Decision from Pre-Appeal Brief Review mailed February 9, 2006.

Claims 2, 17, 38, and 53 also stand rejected under 35 U.S.C. § 102(b), as being anticipated by Dobbins, et al., U.S. Patent No. 5,509,123 (hereinafter referred to as “Dobbins”), as indicated in the Final Office Action mailed August 23, 2005 (hereinafter referred to as “FOA”), the Advisory Action mailed November 14, 2005 (hereinafter referred

to as “AA”), and the Notice of Panel Decision from Pre-Appeal Brief Review mailed February 9, 2006.

ARGUMENT

35 U.S.C. §102(b) Rejection of Claims 1,3 -6, 16, 18-21, 31-37, 39-43, 52, and 54-57

Appellant asserts that claim 1 is not anticipated by Dobbins. Claim 1 recites a method comprising:

receiving at least one packet; and

disposing of the received at least one packet in response to a walk of a Hash Table, wherein

the Hash Table is balanced,

the Hash Table is configured to store Binary Comparison Trees, and

the Hash Table is configured to encode an Access Control List.

“A claim is anticipated only if each and every element as set forth in the claim is found, either expressly or inherently described, in a single prior art reference.” *Verdegall Bros. V. Union Oil Co. of California*, 814 F.2d 628, 631, 2 USPQ2d 1051, 1053 (Fed. Cir. 1987). Appellant asserts that the cited portions of Dobbins do not expressly or inherently describe the “Hash Table” recited in claim 1, which is configured to “store Binary Comparison Trees” as well as “encode an Access Control List.” Accordingly, claim 1 is not anticipated by the cited portions of Dobbins.

Examiner’s Arguments

In the Final Office Action, the Examiner states that either the cache or the AVL tree taught in Dobbins anticipates the “Hash Table” recited in claim 1. FOA, p. 2. Subsequently, when responding to Appellant’s arguments in the Advisory Action, the Examiner changes focus and equates the forwarding table (also referred to as “forward lookup table”) described in Dobbins with the “Hash Table” recited in claim 1. AA, p. 2.

Description of Dobbins

Dobbins provides “an object-oriented architecture for network layer routing.” Dobbins, Abstract. In this architecture, which “utilizes distributed autonomous forwarding engines,” “each interface 11, 14, 17 has a forwarding engine 12, 15, 18 sitting above it, and each forwarding engine knows how to receive and transmit packets on its own interface.” Dobbins, col. 7, lines 31-36. Several sections, particularly sections B.2 and B.3, of Dobbins describe these distributed forwarding engines.

In section B.3, Dobbins describes how the access list is implemented in the AVL tree (which is cited by the Examiner as teaching the “Hash Table” of claim 1): “To provide an object-oriented, powerful and very efficient access control mechanism, a base class FAC (Forwarding Access) was invented. For efficiency, FAC keeps access list entries as nodes in an AVL tree. A tree does not have a predefined size and may grow freely.” Dobbins, col. 11, lines 9-13.

In section B.2, Dobbins describes how each forwarding engine contains a cache (also cited as anticipating the “Hash Table” of claim 1). Dobbins states that: “Performance-sensitive code often employs caching to speed up performance. Typically, hash codes are used to speed retrieval of the hashed data.” Dobbins, col. 9, lines 31-33. “As part of the forwarding process, the IP forwarding engine methods (1) validate packet addresses, (2) filter against an access list, and (3) retrieve the next hop from the FIB. These procedures are inherently slow, so the results of these procedures once obtained, such as address validity, are cached and corresponding procedures are provided in IPACache to lookup the same results quickly.” Dobbins, col. 10, lines 22-28.

In column 7, Dobbins describes how the forwarding engines operate. “Each forwarding engine accesses a common forwarding table 20.” Dobbins, col. 7, lines 39-40. The forwarding engine (also cited as anticipating the “Hash Table” of claim 1) includes a service method that “provides a next-hop determination [for a data packet] by looking up the destination network address in a cache memory of active addresses... and, if the destination network address is not located in cache memory, accessing a forward look-up table 20 for the best route to the destination network address.” Dobbins, col. 7, lines 55-67.

Thus, Dobbins describes a system in which a forwarding engine can access an access list, implemented as an AVL tree, to determine how to filter a packet. Once a filtering decision is made, the forwarding engine can store the result of the filtering procedure in a

cache, which can subsequently be accessed using a hash code. Appellant notes that in the system taught in Dobbins, the information stored in the cache is a single result (e.g., “permit” or “deny”; See Dobbins, col. 10, lines 38-41), obtained by accessing an access list, and is not an access list itself. The forwarding engine can access a forwarding table to determine the best route to use when sending a packet to a destination network address.

Dobbins’ Cache does not anticipate the Hash Table of Claim 1

As noted above, in the Final Office Action, the Examiner equates the cache of Dobbins with the “Hash Table” of claim 1. FOA, p. 2. The descriptions of the cache in the cited portions of Dobbins suggest that “hash codes are used to speed retrieval of the hashed data” within the cache. However, the cache described in these portions of Dobbins is not configured in the same manner as the “Hash Table” of claim 1. In particular, claim 1 states that the “Hash Table is configured to store Binary Comparison Trees.” The cache recited in Dobbins stores the result of a filtering procedure, not a binary comparison tree. As noted above, Dobbins describes the result as being a single permission (e.g., “permit” or “deny”). A single permission such as “permit” or “deny” is clearly not a binary comparison tree. Accordingly, the cache of Dobbins does not anticipate, teach, or suggest a “Hash Table” that is “configured to store Binary Comparison Trees”, as recited in claim 1.

Furthermore, the cache of Dobbins does not teach or suggest a “Hash Table” that is “configured to encode an Access Control List,” as recited in claim 1. As noted above, the cache simply stores results of the filtering procedure. The Examiner has not cited any portion of Dobbins that teaches or suggests that the cache encodes an access control list. Accordingly, the cited descriptions of the cache also fail to anticipate this feature of claim 1.

Dobbins’ AVL Tree does not anticipate the Hash Table of Claim 1

The Examiner also equates the AVL tree described in Dobbins with the “Hash Table” of claim 1. FOA, p. 2. However, the cited portions of Dobbins that describe the AVL tree clearly do not teach or suggest that the AVL tree is a hash table, nor do those portions of Dobbins teach or suggest that the AVL tree be stored within a hash table. An AVL tree is clearly not the same thing as a hash table, and the Examiner has provided no evidence or explanation as to why a description of a tree teaches or suggests a hash table. Accordingly,

based on the plain meaning of the term “Hash Table,” the descriptions of the AVL tree in the cited portions of Dobbins do not teach or suggest the “Hash Table” of claim 1.

The Examiner also maintains that the AVL tree anticipates a Hash Table that “is configured to store Binary Comparison Trees” and “configured to encode an Access Control List.” FOA, p. 2. As noted above, the AVL tree is a tree, not a hash table. No cited portion of Dobbins teaches or suggests that the AVL tree be stored in a hash table. Similarly, no cited portion of Dobbins teaches or suggests that an access control list be encoded in a hash table. Instead, Dobbins teaches that access control lists are kept as nodes in the AVL tree. Dobbins, col. 11, lines 12-13. Accordingly, the cited portions of Dobbins describing the AVL tree clearly neither teach nor suggest a hash table that is configured to store binary comparison trees and to encode an access control list.

Dobbins’ Forwarding Table does not anticipate the Hash Table of Claim 1

When responding to Appellant’s arguments in the Advisory Action, the Examiner equates the “Hash Table” of claim 1 with a “forwarding table or lookup table.” AA, p. 2. The Examiner also cites “forward lookup table 20” in Dobbins as anticipating the “Hash Table.” AA, p. 2. There is no teaching or suggestion in the cited portions of Dobbins that the forward lookup table 20 is implemented as a hash table, and thus the cited description of forward lookup table 20 does not anticipate claim 1.

Additionally, the cited description of the forwarding table does not teach or suggest a hash table that is configured in the same way as the “Hash Table” recited in claim 1. Instead, the cited description simply describes the forwarding table as being a structure that is accessed to determine the best route for a packet, based on the packet’s destination address. Dobbins, col. 7, lines 55-67. The cited description neither teaches nor suggests that the forwarding table is a “Hash Table” configured to “encode an Access Control List” or “store Binary Comparison Trees” as recited in claim 1. This assertion is supported by the Examiner’s continued reliance on Dobbins’ AVL tree to anticipate these features of the “Hash Table” of claim 1. AA, p. 2-3. Accordingly, the cited description of the forwarding table does not anticipate the “Hash Table” of claim 1, which is configured to “store Binary Comparison Trees” as well as to “encode an Access Control List.”

Dobbins does not anticipate the Hash Table of Claim 1

Thus, as indicated above, the portions of Dobbins cited in the rejection of claim 1 do not anticipate the “Hash Table” of claim 1, which is a hash table that is configured to “store Binary Comparison Trees” as well as to “encode an Access Control List.” For at least the foregoing reasons, claim 1 is patentable over the cited art. Claims 2-6 are also patentable for at least these reasons.

Independent claims 16, 35, and 52 each recite a “Hash Table” that is configured to “store Binary Comparison Trees” as well as to “encode an Access Control List.” Accordingly, these claims are patentable for reasons similar to the foregoing reasons provided above with respect to claim 1. Dependent claims 17-21, 31-34, 36-43 and 53-57 are patentable for at least these reasons.

35 U.S.C. §102(b) Rejection of Claims 2, 17, 38, and 53

Appellant asserts that claim 2 is not anticipated by Dobbins. Claim 2 recites a method comprising:

constructing a hash table index value from one or more bit positions, within the
received at least one packet, pointed at by one or more pointers of a Hash-
Table-Balancing Bit Selection Vector; and

walking a binary comparison tree associated with the constructed hash table index
value.

The Examiner relies upon col. 11, lines 19-20 and thereafter to anticipate the features of claim 2 reprinted above. FOA, p. 3. Col. 11, lines 17-23 of Dobbins recite:

Efficiency is further maximized because within each access list, valid entries are also linked across the tree in their sequence order for fast scan during filtering. Each interface may associate with one and only one access list as identified by an ID in the entry. This association is done in the forwarding part of a protocol's MIB via an ID leaf and a control leaf to enable or disable filtering.


Thus, Dobbins teaches that an ID in an entry of an access control list can identify the one and only access list that is associated with an interface. The cited portions of Dobbins do not teach or suggest that the ID (or any other feature described in the cited portion of Dobbins) is in any way constructed from (or even related to) a received packet. Accordingly, this portion of Dobbins clearly neither teaches nor suggests constructing an index value from

one or more bit positions within a received packet. Furthermore, the cited portions of Dobbins do not teach or suggest pointers that point at the bit positions within the received packet. For at least these reasons, the cited portions of Dobbins do not anticipate “constructing a hash table index value from one or more bit positions, within the received at least one packet,” as recited in claim 2. Similarly, the cited portions of Dobbins do not anticipate “one or more bit positions, within the received at least one packet, pointed at by one or more pointers of a Hash-Table-Balancing Bit Selection Vector,” as recited in claim 2.

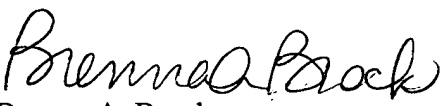
Accordingly, claim 2 is not anticipated by the cited portions of Dobbins. Claims 17, 38, and 53 are similarly patentable over the cited art.

CONCLUSION

For the above reasons, Appellant respectfully submits that the rejections of pending claims 1-6, 16-21, 31-43, and 52-57 are unfounded. Accordingly, Appellant respectfully requests that the Board reverse the rejections of these claims.

I hereby certify that this correspondence is being deposited with the United States Postal Service as First Class Mail in an envelope addressed to: Mail Stop: <u>Appeal Brief - Patents</u> , Commissioner for Patents, P. O. Box 1450, Alexandria, VA 22313-1450, on <u>March 9, 2006</u> .	
 _____ Attorney for Appellant	<u>3-9-2006</u> _____ Date of Signature

Respectfully submitted,


 Brenna A. Brock
 Attorney for Appellant
 Reg. No. 48,509
 512-439-5087 (Telephone)
 512-439-5099 (Facsimile)

CLAIMS APPENDIX

1. (Previously Presented) A method comprising:
receiving at least one packet; and
disposing of the received at least one packet in response to a walk of a Hash Table,
wherein
the Hash Table is balanced,
the Hash Table is configured to store Binary Comparison Trees, and
the Hash Table is configured to encode an Access Control List.
2. (Previously Presented) The method of Claim 1, wherein said disposing of the received at least one packet in response to a walk of the Hash Table further includes:
constructing a hash table index value from one or more bit positions, within the received at least one packet, pointed at by one or more pointers of a Hash-Table-Balancing Bit Selection Vector; and
walking a binary comparison tree associated with the constructed hash table index value.
3. (Previously Presented) The method of Claim 1, further comprising:
converting the Access Control List to the Hash Table.
4. (Previously Presented) The method of Claim 3, wherein said converting the Access Control List to the Hash Table further includes:
creating a binary comparison tree for at least one Access Control List rule in the Access Control List.
5. (Original) The method of Claim 4, wherein said creating a binary comparison tree for at least one Access Control List rule further includes:

creating at least one node, having at least one miss branch and at least one match branch, for at least one packet header field utilized by the at least one Access Control List Rule in the Access Control List.

6. (Previously Presented) The method of Claim 3, wherein said converting the Access Control List to the Hash Table further includes:

inserting at least a part of a binary comparison tree constructed for at least one Access Control List rule into a hash table entry pointed at by a hash table index.

7. (Original) The method of Claim 6, wherein said inserting at least a part of a binary comparison tree constructed for at least one Access Control List rule into a hash table entry pointed at by a hash table index further includes:

generating a hash table index value for the at least one Access Control List rule; and
inserting the at least a part of a binary comparison tree constructed for at least one Access Control List rule into a hash table entry pointed at by a hash table index which is equal to the generated hash table index value.

8. (Original) The method of Claim 7, wherein said inserting the at least a part of a binary comparison tree constructed for at least one Access Control List rule into a hash table entry pointed at by a hash table index which is equal to the generated hash table index value further includes:

inserting, in its entirety, the binary comparison tree constructed for the at least one Access Control List rule into the hash table entry pointed at by the hash table index in response to a determination that no pre-existing binary comparison tree is resident within the hash table entry.

9. (Original) The method of Claim 7, wherein said inserting the at least a part of a binary comparison tree constructed for at least one Access Control List rule into a hash table entry pointed at by a hash table index which is equal to the generated hash table index value further includes:

inserting at least one node of the binary comparison tree constructed for the at least one Access Control List rule into the hash table entry pointed at by the hash table index in response to a determination that a pre-existing binary comparison tree is resident within the hash table entry.

10. (Original) The method of Claim 7, wherein said generating a hash table index value for the at least one Access Control List rule further includes:

constructing the hash table index value from the contents of one or more packet headers utilized by the at least one Access Control List rule in the Access Control List.

11. (Original) The method of Claim 10, wherein said constructing the hash table index value from the contents of one or more packet headers utilized by the at least one Access Control List rule in the Access Control List further includes:

constructing the hash table index value from the contents of the one or more packet header bit positions pointed at by one or more pointers of a Hash-Table-Balancing Bit Selection Vector.

12. (Original) The method of Claim 11, wherein said constructing the hash table index value from the contents of the one or more packet header bit positions pointed at by one or more pointers of a Hash-Table-Balancing Bit Selection Vector further includes:

constructing the Hash-Table-Balancing Bit Selection Vector.

13. (Original) The method of Claim 12, wherein said constructing the Hash-Table-Balancing Bit Selection Vector further includes:

defining one or more pointers of the Hash-Table-Balancing Bit Selection Vector to point to one or more bit positions in one or more packet header fields utilized by one or more rules of the Access Control List.

14. (Original) The method of Claim 13, wherein said defining one or more pointers of the Hash-Table-Balancing Bit Selection Vector to point to one or more bit positions in one or more packet header fields utilized by one or more rules of the Access Control List further includes:

defining the one or more pointers of the Hash-Table-Balancing Bit Selection Vector to point to one or more bit positions, which appear relatively most frequently, within the one or more packet header fields utilized by the one or more Rules of the Access Control List.

15. (Original) The method of Claim 13, wherein said defining one or more pointers of the Hash-Table-Balancing Bit Selection Vector to point to one or more bit positions in one or more packet header fields utilized by one or more rules of the Access Control List further includes:

defining the one or more pointers of the Hash-Table-Balancing Bit Selection Vector to point to one or more bit positions, whose contents have relatively equal variation between logical one and logical zero, within the one or more packet header fields utilized by the one or more Rules of the Access Control List.

16. (Previously Presented) A system comprising:

means for receiving at least one packet; and

means for disposing of the received at least one packet in response to a walk of a Hash Table, wherein

the Hash Table is balanced,

the Hash Table is configured to store Binary Comparison Trees, and

the Hash Table is configured to encode an Access Control List.

17. (Previously Presented) The system of Claim 16, wherein said means for disposing of the received at least one packet in response to a walk of a Hash Table further includes:

means for constructing a hash table index value from one or more bit positions,
within the received at least one packet, pointed at by one or more pointers of a
Hash-Table-Balancing Bit Selection Vector; and

means for walking a binary comparison tree associated with the constructed hash
table index value.

18. (Previously Presented) The system of Claim 16, further comprising:

means for converting the Access Control List to the Hash Table.

19. (Previously Presented) The system of Claim 18, wherein said means for
converting the Access Control List to the Hash Table further includes:

means for creating a binary comparison tree for at least one Access Control List rule
in the Access Control List.

20. (Original) The system of Claim 19, wherein said means for creating a binary
comparison tree for at least one Access Control List rule further includes:

means for creating at least one node, having at least one miss branch and at least one
match branch, for at least one packet header field utilized by the at least one
Access Control List rule in the Access Control List.

21. (Previously Presented) The system of Claim 18, wherein said means for
converting the Access Control List to the Hash Table further includes:

means for inserting at least a part of a binary comparison tree constructed for at least
one Access Control List rule into a hash table entry pointed at by a hash table
index.

22. (Original) The system of Claim 21, wherein said means for inserting at least a
part of a binary comparison tree constructed for at least one Access Control List rule into a
hash table entry pointed at by a hash table index further includes:

means for generating a hash table index value for the at least one Access Control List rule; and

means for inserting the at least a part of a binary comparison tree constructed for at least one Access Control List rule into a hash table entry pointed at by a hash table index which is equal to the generated hash table index value.

23. (Original) The system of Claim 22, wherein said means for inserting the at least a part of a binary comparison tree constructed for at least one Access Control List rule into a hash table entry pointed at by a hash table index which is equal to the generated hash table index value further includes:

means for inserting, in its entirety, the binary comparison tree constructed for the at least one Access Control List Rule into the hash table entry pointed at by the hash table index in response to a determination that no pre-existing binary comparison tree is resident within the hash table entry.

24. (Original) The system of Claim 22, wherein said means for inserting the at least a part of a binary comparison tree constructed for at least one Access Control List rule into a hash table entry pointed at by a hash table index which is equal to the generated hash table index value further includes:

means for inserting at least one node of the binary comparison tree constructed for the at least one Access Control List rule into the hash table entry pointed at by the hash table index in response to a determination that a pre-existing binary comparison tree is resident within the hash table entry.

25. (Original) The system of Claim 22, wherein said means for generating a hash table index value for the at least one Access Control List rule further includes:

means for constructing the hash table index value from the contents of one or more packet headers utilized by the at least one Access Control List rule in the Access Control List.

26. (Original) The system of Claim 25, wherein said means for constructing the hash table index value from the contents of one or more packet headers utilized by the at least one Access Control List rule in the Access Control List further includes:

means for constructing the hash table index value from the contents of the one or more packet header bit positions pointed at by one or more pointers of a Hash-Table-Balancing Bit Selection Vector.

27. (Original) The system of Claim 26, wherein said means for constructing the hash table index value from the contents of the one or more packet header bit positions pointed at by one or more pointers of a Hash-Table-Balancing Bit Selection Vector further includes:

means for constructing the Hash-Table-Balancing Bit Selection Vector.

28. (Original) The system of Claim 27, wherein said means for constructing the Hash-Table-Balancing Bit Selection Vector further includes:

means for defining one or more pointers of the Hash-Table-Balancing Bit Selection Vector to point to one or more bit positions in one or more packet header fields utilized by one or more rules of the Access Control List.

29. (Original) The system of Claim 28, wherein said means for defining one or more pointers of the Hash-Table-Balancing Bit Selection Vector to point to one or more bit positions in one or more packet header fields utilized by one or more rules of the Access Control List further includes:

means for defining the one or more pointers of the Hash-Table-Balancing Bit Selection Vector to point to one or more bit positions, which appear relatively most frequently, within the one or more packet header fields utilized by the one or more Rules of the Access Control List.

30. (Original) The system of Claim 29, wherein said means for defining one or more pointers of the Hash-Table-Balancing Bit Selection Vector to point to one or more bit

positions in one or more packet header fields utilized by one or more rules of the Access Control List further includes:

means for defining the one or more pointers of the Hash-Table-Balancing Bit

Selection Vector to point to one or more bit positions, whose contents have relatively equal variation between logical one and logical zero, within the one or more packet header fields utilized by the one or more Rules of the Access Control List.

31. (Previously Presented) The system of Claim 16, further comprising:

signal bearing media bearing

said means for receiving at least one packet, and

said means for disposing of the received at least one packet in response to a walk of the Hash Table.

32. (Original) The system of Claim 31, wherein said signal bearing media further includes:

recordable media.

33. (Original) The system of Claim 31, wherein said signal bearing media further includes:

transmission media.

34. (Original) The system of Claim 16, wherein the system further includes:
a network station.

35. (Previously Presented) A program product comprising:
signal bearing media bearing

means for receiving at least one packet, and

means for disposing of the received at least one packet in response to a walk of a Hash Table, wherein
the Hash Table is balanced,
the Hash Table is configured to store Binary Comparison Trees, and
the Hash Table is configured to encode an Access Control List.

36. (Original) The program product of Claim 35, wherein said signal bearing media further includes:
recordable media.

37. (Original) The program product of Claim 35, wherein said signal bearing media further includes:
transmission media.

38. (Previously Presented) The program product of Claim 35, wherein said means for disposing of the received at least one packet in response to a walk of a Hash Table further includes:

means for constructing a hash table index value from one or more bit positions,
within the received at least one packet, pointed at by one or more pointers of a Hash-Table-Balancing Bit Selection Vector; and
means for walking a binary comparison tree associated with the constructed hash table index value.

39. (Previously Presented) The program product of Claim 35, wherein said signal bearing media bears:
means for converting the Access Control List to the Hash Table.

40. (Previously Presented) The program product of Claim 39, wherein said means for converting the Access Control List to the Hash Table further includes:

means for creating a binary comparison tree for at least one Access Control List rule in the Access Control List.

41. (Original) The program product of Claim 40, wherein said means for creating a binary comparison tree for at least one Access Control List rule further includes:

means for creating at least one node, having at least one miss branch and at least one match branch, for at least one packet header field utilized by the at least one Access Control List rule in the Access Control List.

42. (Previously Presented) The program product of Claim 39, wherein said means for converting the Access Control List to the Hash Table further includes:

means for inserting at least a part of a binary comparison tree constructed for at least one Access Control List rule into a hash table entry pointed at by a hash table index.

43. (Original) The program product of Claim 42, wherein said means for inserting at least a part of a binary comparison tree constructed for at least one Access Control List rule into a hash table entry pointed at by a hash table index further includes:

means for generating a hash table index value for the at least one Access Control List rule; and

means for inserting the at least a part of a binary comparison tree constructed for at least one Access Control List rule into a hash table entry pointed at by a hash table index which is equal to the generated hash table index value.

44. (Original) The program product of Claim 43, wherein said means for inserting the at least a part of a binary comparison tree constructed for at least one Access Control List rule into a hash table entry pointed at by a hash table index which is equal to the generated hash table index value further includes:

means for inserting, in its entirety, the binary comparison tree constructed for the at least one Access Control List Rule into the hash table entry pointed at by the hash table index in response to a determination that no pre-existing binary comparison tree is resident within the hash table entry.

45. (Original) The program product of Claim 43, wherein said means for inserting the at least a part of a binary comparison tree constructed for at least one Access Control List rule into a hash table entry pointed at by a hash table index which is equal to the generated hash table index value further includes:

means for inserting at least one node of the binary comparison tree constructed for the at least one Access Control List rule into the hash table entry pointed at by the hash table index in response to a determination that a pre-existing binary comparison tree is resident within the hash table entry.

46. (Original) The program product of Claim 43, wherein said means for generating a hash table index value for the at least one Access Control List rule further includes:

means for constructing the hash table index value from the contents of one or more packet headers utilized by the at least one Access Control List rule in the Access Control List.

47. (Original) The program product of Claim 46, wherein said means for constructing the hash table index value from the contents of one or more packet headers utilized by the at least one Access Control List rule in the Access Control List further includes:

means for constructing the hash table index value from the contents of the one or more packet header bit positions pointed at by one or more pointers of a Hash-Table-Balancing Bit Selection Vector.

48. (Original) The program product of Claim 47, wherein said means for constructing the hash table index value from the contents of the one or more packet header bit positions pointed at by one or more pointers of a Hash-Table-Balancing Bit Selection Vector further includes:

means for constructing the Hash-Table-Balancing Bit Selection Vector.

49. (Original) The program product of Claim 48, wherein said means for constructing the Hash-Table-Balancing Bit Selection Vector further includes:

means for defining one or more pointers of the Hash-Table-Balancing Bit Selection Vector to point to one or more bit positions in one or more packet header fields utilized by one or more rules of the Access Control List.

50. (Original) The program product of Claim 49, wherein said means for defining one or more pointers of the Hash-Table-Balancing Bit Selection Vector to point to one or more bit positions in one or more packet header fields utilized by one or more rules of the Access Control List further includes:

means for defining the one or more pointers of the Hash-Table-Balancing Bit Selection Vector to point to one or more bit positions, which appear relatively most frequently, within the one or more packet header fields utilized by the one or more Rules of the Access Control List.

51. (Original) The program product of Claim 50, wherein said means for defining one or more pointers of the Hash-Table-Balancing Bit Selection Vector to point to one or more bit positions in one or more packet header fields utilized by one or more rules of the Access Control List further includes:

means for defining the one or more pointers of the Hash-Table-Balancing Bit Selection Vector to point to one or more bit positions, whose contents have relatively equal variation between logical one and logical zero, within the one or more packet header fields utilized by the one or more Rules of the Access Control List.

52. (Previously Presented) A network station comprising:

a per-packet processing engine; and

a hash table, wherein

the hash table is balanced,

the hash table is configured to store binary comparison trees,

the hash table is configured to encode an access control list, and

the per-packet processing engine is configured to walk the hash table, in response to the network station receiving a packet, and

the walk of the hash table identifies a disposition of the packet, according to the access control list encoded in the hash table.

53. (Previously Presented) The network station of Claim 52, wherein the per-packet processing engine is configured to:

construct a hash table index value from one or more bit positions, within the packet, pointed at by one or more pointers of a hash-table-balancing bit selection vector; and

walk a binary comparison tree, stored in the hash table, associated with the constructed hash table index value.

54. (Previously Presented) The network station of Claim 52, further comprising: program instructions executable to convert the access control list to the hash table.

55. (Previously Presented) The network station of Claim 54, wherein the program instructions are is further executable to:

create a binary comparison tree for at least one access control List rule in the Access Control List.

56. (Previously Presented) The network station of Claim 54, wherein the program instructions are further executable to:

insert at least a part of a binary comparison tree constructed for at least one access control list rule into a hash table entry pointed at by a hash table index.

57. (Previously Presented) The network station of Claim 56, wherein inserting at least a part of a binary comparison tree constructed for at least one access control list rule into a hash table entry pointed at by a hash table index includes:

generating a hash table index value for the at least one access control list rule; and

inserting the at least a part of a binary comparison tree constructed for at least one access control list rule into a hash table entry pointed at by a hash table index which is equal to the generated hash table index value.

58. (Previously Presented) The network station of Claim 57, wherein inserting the at least a part of a binary comparison tree constructed for at least one access control list rule into a hash table entry pointed at by a hash table index which is equal to the generated hash table index value includes:

inserting, in its entirety, the binary comparison tree constructed for the at least one access control list rule into the hash table entry pointed at by the hash table index in response to a determination that no pre-existing binary comparison tree is resident within the hash table entry.

59. (Previously Presented) The network station of Claim 57, wherein inserting the at least a part of a binary comparison tree constructed for at least one access control list rule into a hash table entry pointed at by a hash table index which is equal to the generated hash table index value includes:

inserting at least one node of the binary comparison tree constructed for the at least one access control list rule into the hash table entry pointed at by the hash table index in response to a determination that a pre-existing binary comparison tree is resident within the hash table entry.

60. (Previously Presented) The network station of Claim 57, wherein generating a hash table index value for the at least one access control list rule further includes:

constructing the hash table index value from the contents of one or more packet headers utilized by the at least one access control list rule in the access control list.

61. (Previously Presented) The network station of Claim 60, wherein constructing the hash table index value from the contents of one or more packet headers utilized by the at least one access control list rule in the access control list includes:

constructing the hash table index value from the contents of the one or more packet header bit positions pointed at by one or more pointers of a hash-table-balancing bit selection vector.

62. (Previously Presented) The network station of Claim 62, wherein said constructing the hash table index value from the contents of the one or more packet header bit positions pointed at by one or more pointers of a hash-table-balancing bit selection vector includes:

defining one or more pointers of the hash-table-balancing bit selection vector to point to one or more bit positions in one or more packet header fields utilized by one or more rules of the access control list.

63. (Previously Presented) The network station of Claim 62, wherein defining one or more pointers of the hash-table-balancing bit selection vector to point to one or more bit positions in one or more packet header fields utilized by one or more rules of the access control list includes:

defining the one or more pointers of the hash-table-balancing bit selection vector to point to one or more bit positions, which appear relatively most frequently, within the one or more packet header fields utilized by the one or more rules of the access control list.

64. (Previously Presented) The network station of Claim 62, wherein defining one or more pointers of the hash-table-balancing bit selection vector to point to one or more bit positions in one or more packet header fields utilized by one or more rules of the access control list includes:

defining the one or more pointers of the hash-table-balancing bit selection vector to point to one or more bit positions, whose contents have relatively equal variation between logical one and logical zero, within the one or more packet header fields utilized by the one or more rules of the access control list.

EVIDENCE APPENDIX

None

RELATED PROCEEDINGS APPENDIX

None